数据分块算法在定位差异数据时的作用分析

黄文豪[1,2],齐德昱[1,2],谢　嵘[1],刘　宇[1],张皓同[1,2]
(1. 广东外语外贸大学南国商学院 数字化技术研究院,广东 广州 510545;
2. 华南理工大学 计算机科学与工程学院,广东 广州 510641)

摘　要:差异数据定位在数据增量同步等领域得到了很好的应用。当下学者们将数据分块算法应用在差异数据定位中,提出了很多优秀的数据分块算法,并对各自算法的效率进行了论述和实验,但并没有从理论上论述数据分块算法在差异数据定位中的正确性。此外,定位到的差异数据的大小与数据分块算法的关系也没有进行理论分析,多是以实验结果来辅助说明。为此,文中对数据差异定位的过程进行抽象,对数据分块算法应用于该过程的正确性进行了论证,并对数据分块算法在数据差异定位中的作用进行分析。通过理论推导的方式,证明了数据分块算法在定位差异数据时的正确性,同时得出差异数据的大小与数据分块算法的关系。文中结论对设计应用于差异数据定位的数据分块算法有一定的参考意义。

关键词:数据分块算法;差异数据定位;理论分析;数据增量同步;逻辑推理
中图分类号:TP311　　**文献标识码**:A　　**文章编号**:1673-629X(2023)10-0022-06
doi:10.3969/j.issn.1673-629X.2023.10.004

Analysis of Function of Data Chunking Algorithm in Locating Delta Data

HUANG Wen-hao[1,2],QI De-yu[1,2],XIE Rong[1],LIU Yu[1],ZHANG Hao-tong[1,2]
(1. Digital Technology Research Institute,Guangdong University of Foreign Studies
South China Business College,Guangzhou 510545,China;
2. School of Computer Science and Engineering,South China University of Technology,Guangzhou 510641,China)

Abstract:Delta data location has been well applied in data incremental synchronization and other fields. At present,scholars have applied data chunking algorithms in delta data location,put forward many excellent data chunking algorithms,and discussed and experimented on the efficiency of each algorithm, but did not discuss the correctness of data chunking algorithm in delta data location theoretically. Besides,there is no theoretical analysis on the relationship between the size of the delta data and the data chunking algorithm,and most of them are only explained by the experimental results. Therefore,we abstract the process of delta data location,demonstrate the correctness of the data chunking algorithm applied in the process and analyze the role of the data chunking algorithm in delta data location. Through theoretical derivation,the correctness of the data chunking algorithm in locating delta data is proved,and the relationship between the size of delta data and the data chunking algorithm is obtained. The conclusion has a certain reference significance for the design of data chunking algorithm applied to delta data location.

Key words:data chunking algorithm;differential data location;theoretical analysis;data delta synchronization;logical reasoning

0　引　言

随着数据量的快速增长,如何定位两个相似数据之间的差异数据,即差异数据定位,也得到了快速的发展。在现有的研究中,学者们多采取数据分块的方式来找出两个相似数据间不同的分块,这些分块则是差异数据。这种方式的工作过程如图 1 所示。具体的,首先对 $Data_1$ 和 $Data_2$ 按照相同的数据分块算法进行分块,然后比较两组分块之间不同的分块,这些不同的分块所组成的数据即为差异数据。

学术界关于差异数据分块的研究重点集中在数据分块算法的设计上。数据分块算法的不同将直接影响差异数据定位的效果,比如能否全部定位出不同的数据、定位到的差异数据中相同数据的多少等。

不过学术界提出的数据分块算法都是在实验上验

收稿日期:2022-11-19　　**修回日期**:2023-03-22
基金项目:国家自然科学基金项目(61070015);广州市产业技术重大攻关计划项目(201802020035)
作者简介:黄文豪(1988-),男,博士生在读,通信作者,研究方向为大数据、人工智能、深度学习和软件工程;齐德昱(1959-),男,教授,研究方向为软件开发方法和体系结构、软件开发环境和工具等;谢　嵘(1974-),男,副教授,研究方向为网络安全、无线网络、移动计算。

证算法在差异数据定位中起到的作用,并没有从理论上给出数据分块算法能否定位出所有不同的数据,也没有探讨数据分块算法中各参数在差异定位中所起到的作用。因此,该文对差异数据定位的过程进行抽象,推导出了这一过程的正确性。同时,还对数据分块算法中参数与差异数据定位的关系进行了讨论。

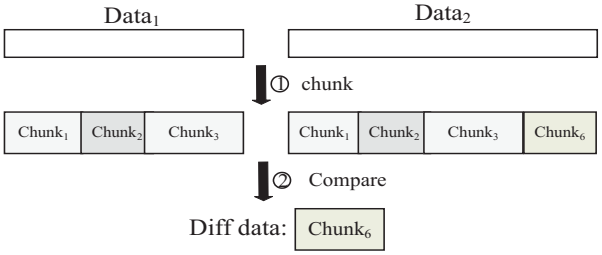

图 1　差异数据定位的过程

1　相关工作

借助数据分块算法来定位差异数据这一方法多用在重复数据删除、数据增量同步等领域。数据分块算法是对数据进行分块,通过对每个块进行处理,解决数据整体不易处理的问题,是一种化整为零的思路。该文探讨的数据差异定位主要是为了更好地应用于数据增量同步,因为重复数据删除对分块的稳定性会有额外的要求,不过该文得到的一些结论也可以在重复数据删除中起到借鉴作用。

数据分块算法最早应用在差异数据定位中是 A. Tridgell 提出的一种基于多轮通讯的同步算法 Rsync[1],该算法中利用数据分块算法对需要同步的文件进行等长分块,然后通过分块比较得到两个文件间的差异数据。不过 Rsync 中使用的数据分块算法是固定长度的数据分块算法,这类算法存在字节漂移的问题:当在文件的起始位置插入一个字节,将会导致文件所有的分块发生变化。这种问题在定位差异数据时会导致发现的差异数据远大于实际变化的数据,虽然 Rsync 采用了别的方法规避了字节漂移,但却大大增加了算法的计算量。在 Rsync 的基础上,为了加快差异定位的速度,Lkhagvasuren Ider 等人提出了一种两级分块策略[2],数据块大小分别为 4 MB 和 32 KB。在第一级,使用 4 MB 块大小的索引表,采用字节索引分块的方法快速检测出大尺寸相同的数据块。在第二级,使用 32 KB 索引表对通过第一级文件相似性检测生成的整个非重复数据区域执行字节索引分块。

为了避免固定长度分块算法的字节漂移问题,学者们开始探索可变长度分块算法(Content – Defined Chunking,CDC)在差异数据定位中的应用。最早的可变长度分块算法基于 Rabin 指纹[3]的分块算法,该算法以匹配指纹为分块依据,增加了分块的抗字节漂移能力。Xia Wen 等人为了追求分块速度,提出了 FastCDC 算法[4],在指纹匹配过程中使用 Gear 指纹代替 Rabin 指纹,并在窗口移动过程中使用一次移动两个字节的方式加快文件遍历速度,同时还使用两种指纹匹配难度增加了分块的稳定性。文章中,作者对 FastCDC 在单次匹配过程中成功的概率进行了简单的介绍,并讨论了跨字节时匹配概率的变化,但没有推敲算法设计上的优劣。为了减少指纹匹配存在的较大计算量,Bjrner Nikolaj 等人提出了基于区间最大值的分块算法 LMC[5],通过寻找一个固定窗口内的最大值是否在窗口的中间位置来判断是否设置切点,文中探讨了 LMC 算法在寻找切点时的概率问题,但同样没有讨论这个概率对差异定位的影响。除此之外,还有一些应用在重复数据删除[6-7]中的数据分块算法[8-11],这些算法的作者在论文中也有提及算法的分块效率、分块平均长度等,但都没有讨论数据分块算法各参数对差异数据定位的影响。

已有的研究中,学者们提出了很多数据分块算法以提升差异数据定位的效果,比如速度[12-13]、差异数据的大小[14-16]。但是这些研究是以实验数据为支撑,通过与已有算法做比较,得到实验上的优势,然后证明所提算法的先进性。通过实验的手段来验证算法的先进性会因为实验数据或算法参数的不同而产生偏差,且没有去探讨借助数据分块算法来定位差异数据的正确性,比如能否定位到所有的不一样的数据。此外,定位到的差异数据大小与数据分块算法存在什么关系,怎样设计数据分块算法才能在包含所有不一样的数据的前提下尽可能地减少定位到的差异数据,这些问题也是现有研究中的空缺。因此,为了解决这些问题,该文将数据分块算法抽象成一种窗口的模式匹配过程,并在此基础上证明了借助数据分块算法来定位差异数据的正确性,同时给出了定位到差异数据大小与数据分块算法的关系,为后人在设计数据分块算法时提供参考。

2　准备知识

本节对差异数据定位的详细过程进行阐述,并将这一过程中涉及到的对象进行符号定义以便后文讨论。同时,对数据分块算法进行了抽象,并对其中涉及到的参数等进行了符号定义,以方便后文的讨论。

在定位差异数据时,使用数据分块算法对 $data_1$ 和 $data_2$ 进行分块。数据分块算法的目的是对数据进行分块,分块的依据是在数据中寻找切点,相邻两个切点之间的数据就属于一个分块。数据分块算法可以分为固定长度分块和基于内容的数据分块。固定长度分块由于存在字节漂移的问题,在定位差异数据时一般不采

用。基于内容的数据分块算法(Content – Defined Chunking,CDC)是在待分块数据中寻找符合特定条件的数据窗口,首先从数据的第一个字节开始,选取窗口大小的相邻数据,如果符合特定条件,则在该数据窗口的最后一个字节处设置切点,然后从下一个字节开始选取窗口大小的相邻数据,继续寻找切点。如果不符合,则将窗口往后移动一个字节,继续判断,直至找到符合特定条件的窗口或者数据结束。

该文将寻找符合特定条件的数据窗口的过程抽象为基于固定长度窗口的模式匹配,并记模式匹配成功的概率记为 θ_1。在对已有数据分块算法的研究中发现,大多数数据分块算法寻找切点的方式符合该抽象模型。在对以往 CDC 的分析中得到一个有趣的结论,即对于一个给定的 CDC,在一次随机匹配的过程中,θ 为一个可计算的固定值。此外,相邻窗口匹配成功的概率是不相互独立的,这是因为在进行模式匹配的过程中是采取逐字节移动的方式,因此相邻窗口间会存在数据的重叠,如图 2 所示。由于 $Window_1$,$Window_2$ 之间存在重叠的数据,$Window_1$ 匹配成功有可能增加 $Window_2$ 匹配成功的概率,反之亦然。但是为了更加客观地对待匹配的随机性同时降低推导的难度,该文假定所有窗口进行模式匹配时成功或失败是相互独立的。

图 2 重叠窗口的情况

基于固定长度窗口的模式匹配过程如图 3 所示。对于窗口(图 3 中的 Window)内的数据,如果满足预先设计的匹配规则,则视为匹配成功,反之匹配失败。需要注意的是,匹配成功并不一定会在窗口处形成分块的切点,比如图 3 中,假设 $Window_1$ 处形成了一个切点,那么即使 $Window_2$ 可以匹配成功也不会产生切点,这是因为下一次的模式匹配是从字节 A3 开始的。

图 3 基于固定长度窗口的模式匹配过程

为了方便后文的讨论,对用到的一些对象或者名称进行符号定义。

$data_1$:定位差异数据时的原数据。

$data_2$:定位差异数据时,在 $data_1$ 的基础上修改之后得到的数据。

n:模式匹配窗口的长度。

θ:一个随机窗口进行模式匹配时成功的概率。

FW(Fit Window):模式匹配成功且形成切点的数据窗口。

3 差异数据定位正确性与数据分块算法的关系

本节讨论借助数据分块算法来定位差异数据时的正确性。通过给出一些定义,并借助定义来推导差异数据定位的正确性。

定义 1:字节数据是一个集合,集合中的元素满足条件:由 1 个或多个字节按照特定顺序拼接而成数据。字节数据记作 B。对于 $b \in B$,b 中的每个字节 byte,记作 $\text{byte} \mapsto b$。

在存储领域,任意一个数据均可以视为 B 的一个元素。如果将完全相同的两个数据视为一个,那么数据与 B 中元素是一一对应的关系。对存储在磁盘上的数据进行处理时,默认不会发生磁盘故障。基于这一点,在讨论的过程中,对于发生概率小于磁盘故障概率的事件,视为不会发生。

定义 2:$<=>$,表示 B 上的一个二元关系。对于 $\forall b_1, b_2 \in B$,当满足 b_1 与 b_2 相同时,记作 $b_1 <=> b_2$,反之记作 $b_1 <\neq> b_2$。

引理 1:对于 $\forall b_1, b_2 \in B$,如果 b_1 与 b_2 不同的概率 $p < \theta$,其中 θ 表示磁盘损坏的概率,则 $b_1 <\neq> b_2$。

证明:因对存储在磁盘上的数据进行讨论时,发生概率小于 θ 的事件视为不会发生,因此依然存在 b_1 与 b_2 相同,即 $b_1 <=> b_2$。

定义 3:对于 $\forall b \in B$,如果存在 B 上的一个映射关系 f,使得唯一存在 $\forall b' \in B$,且 $f(b) <=> b'$,则称 f 为 B 的一个属性,$f(b)$ 为 b 在该属性上的值。

定义 4:对于 $\forall b_1, b_2 \in B$,f 为 B 的一个属性,如果 $f(b_1) <=> f(b_2) \rightarrow b_1 <=> b_2$,则称 f 为 B 的可信属性。

引理 2:对于 $\forall b_1, b_2 \in B$,f 为 B 的一个可信属性,则 $f(b_1) <\neq> f(b_2) \rightarrow b_1 <\neq> b_2$。

证明:反证法,假设存在 $b_1 <\neq> b_2 \wedge f(b_1) <=> f(b_2)$,这与可信属性的定义矛盾,引理得证。

引理 3:MD5 哈希是 B 的可信属性。

证明:对于 $\forall b \in B$,其 MD5 哈希存在唯一性,即 MD5 哈希是 B 的一个属性。对于 MD5 相同的两个数据,它们不同的概率低于磁盘故障的概率,由引理 1 及可信属性的定义可知,MD5 哈希为 B 的可信属性。引理得证。

接下来,将定位差异数据的过程换一种描述方式:对 $data_1$ 进行分块得到分块集合 $B_1 \subseteq B$,对 $data_2$ 按照相同的方式分块得到分块集合 $B_2 \subseteq B$;对 B_1,B_2 计算哈希值得到 B_1',$B_2' \subseteq B$;令集合 $B_3' = B_1' - B_2'$,则 B_3' 中所有哈希值对应的分块集合 $B_3 \subseteq B$,B_3 即为差异数据,记作 BD。

定理 1:对于 $\forall byte \mapsto data_1$, $\exists bd \in BD$, s.t.!$(byte \mapsto data_2) \rightarrow byte \mapsto bd$

证明:在对 $data_1$ 进行分块时,假设 byte 分在 chunk $\in B$ 中。由于!$(byte \mapsto data_2)$,则对 $data_2$ 进行分块时不存在与 chunk 相同的块。由引理 2 可知,B_3' 中包含 chunk 的哈希值,因此 chunk $\in$ BD。定理得证。

由定理 1 可知,对于存在于 $data_1$ 中的字节,如果不存在于 $data_2$ 中,则一定会被检测到。不过,在实际应用中,单检测出差异的字节是不够的,还需要根据 $data_1$ 和 BD 来得到 $data_2$。

定理 2:对于 $\forall data_1, data_2 \in B$,设它们之间的差异数据为 BD,若 $data_1$ 的分块集合为 B_1,$data_2$ 的分块集合为 B_2,则 $B_2 = B_1 + BD$。

证明:反证法,假设 $\exists b$,s.t $b \in B_2 \& b \notin B_1 \& b \notin BD$。由于 $b \in B_2 \& b \notin B_1$,由差异数据的定位过程可知 $b \in BD$,这与假设相矛盾,定理得证。

由定理 2 可知,B_2 中所有分块均可以从 B_1 或 BD 的分块中获得。因此,借助数据分块算法来实现差异数据定位时,只需要记录 B_2 中分块在 B_1 或 BD 中对应的分块,就可以在 B_1 所在的节点,结合 BD 来拼接出 B_2。这就意味着,当 B_1 和 B_2 不在同一节点上时,B_1 只需要 BD 就可以得到 B_2 的内容,进而可以实现增量同步等目的。

综合以上讨论,使用数据分块算法来实现差异数据定位不仅可以保证差异数据中包含了所有的变化数据,还可以借助 $data_1$ 和差异数据来获得 $data_2$。

4　差异数据大小与数据分块算法的关系

在理想的情况下,差异数据的大小理论上应当与实际的修改量相同。当使用数据分块算法来定位差异数据时,由于最小单位是分块,而不是字节,因此定位出的差异数据往往比实际修改量大。为此,该文将对实际得到的差异数据大小进行讨论。

如上文所说,$data_2$ 是在 $data_1$ 的基础上修改得到的,那么寻找 $data_2$ 和 $data_1$ 之间的差异数据实际上就是寻找在对 $data_1$ 的一次修改后受影响的分块。数据的修改一般包括三种:增加、删除和变动。数据的修改量也可以有很多种情况,该文首先以单个字节的变动为例来讨论受影响的分块。假设 $data_1$ 的分块结果如图 4 所示。其中 FW_i 表示匹配成功的窗口。设模式匹配窗口的长度为 n,模式匹配成功的概率为 θ,$data_1$ 为无边界的一段数据,其长度为 L。

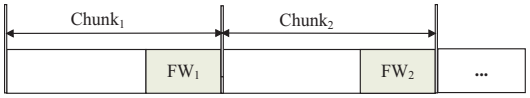

图 4　$data_1$ 的分块结果

当单个字节变动发生后,会有两种情况发生:一种是包含该字节的某个窗口形成了 FW,另一种是包含该字节的所有窗口都没有形成 FW。如果要 $data_1$ 在字节变动后受影响的分块,就必须找到受影响的 FW_i。如果原来的 FW_i 所在的窗口依然是 FW,那么受影响的分块就是字节变动所在的分块。如果原来的 FW_i 所在的窗口不再是 FW,那么受影响的分块就是 FW_i 所在的分块以及相邻的下一个分块,小概率影响更多后续分块(FW_i 之间距离较近的情况)。接下来,在这两种情况下分别讨论对 FW_i 的影响。在讨论之前,需要有一个假设:不存在两个满足匹配条件的窗口有重叠区域,否则会产生严重的字节漂移问题。

当包含该字节的某个窗口形成了 FW 时(只会有一个窗口形成 FW),会有两种情况使得原来的 FW_i 依然是 FW:一是原来的 FW_i 在新形成的 FW 之前或两者不存在重叠区域,一种是原来的 FW_i 与新形成的 FW 完全重叠。当原来的 FW_i 在新形成的 FW 之前时,即使存在重叠区域,也会将新形成的 FW 破坏掉。设包含该字节的某个窗口形成了 FW 时,原来的 FW_i 依然是 FW 的概率为 P_1,则:

$$P_1 = (1-\theta)^{n-1} \tag{1}$$

设包含该字节的某个窗口形成了 FW 的概率为 P_2,则:

$$P_2 = 1 - (1-\theta)^n \tag{2}$$

当包含该字节的所有窗口都没有形成 FW 时,概率记为 P_3。会有一种情况使得原来的 FW_i 依然是 FW:该字节不在 FW_i 中。设包含该字节的所有窗口都没有形成 FW 时,原来的 FW_i 依然是 FW 的概率为 P_4。则:

$$P_3 = 1 - P_2 \tag{3}$$

$$P_4 = (1-\theta)^n \tag{4}$$

设一次字节变动所引起的差异数据为,则可以得出公式(5)。

$$N = \sum_{i=1}^{L} \frac{1}{L}(P_2P_1x_i + P_2(1-P_1)(x_i + x_{i+1} + \triangle) + P_3P_4x_i + P_3(1-P_4)(x_i + x_{i+1} + \nabla)) \tag{5}$$

$$N = P_2 * 2E(x_i) - P_2P_1E(x_i) + P_3 * 2E(x_i) - P_3P_4E(x_i) + \nabla \tag{6}$$

$$N = (\frac{1}{\theta} + n - 1)(2 + \theta(1-\theta)^{2n-1} - (1-\theta)^{n-1} + \nabla) \tag{7}$$

$$\frac{\partial N}{\partial n} = 2 + \theta(1-\theta)^{2n-1} - (1-\theta)^{n-1} + (\frac{1}{\theta} + n - 1)(2\theta(1-\theta)^{2n-1}\ln(1-\theta) - (1-\theta)^{n-1}\ln(1-\theta)) \tag{8}$$

$$N = \frac{1}{\theta}(2 + \theta(1-\theta) - 1) + \nabla =$$

$$\theta^{-1} - \theta + 1 + \nabla \tag{9}$$

式中，i 表示 $data_1$ 中的字节下标，x_i 表示 FW_i 所在分块的长度，因此下标 i 对应的字节是属于 x_i 的，∇ 是一个极小值。根据模式匹配的方式，可以得出 $(x_i - n + 1) \sim GE(\theta)$，当 L 趋于无穷大时，x_i 可以取到期望值 $E(x_i) = \frac{1}{\theta} + n - 1$。对公式(5)化简得到公式(6)，对公式(6)进行化简计算得公式(7)。

为了探讨 n 对 N 的影响，计算 N 对 n 的偏微分如公式(8)所示，其中 n 为正整数，$0 < \theta < 1$。因为 $\theta(1-\theta)^{2n-1} < 1$，$(1-\theta)^{n-1} < 1$，所以 $2 + \theta(1-\theta)^{2n-1} - (1-\theta)^{n-1} > 0$。接下来对公式(8)的后半部分进行讨论，令 $t = (\frac{1}{\theta} + n - 1)(2\theta(1-\theta)^{2n-1}\ln(1-\theta) - (1-\theta)^{n-1}\ln(1-\theta))$，需要计算 t 的正负，进而可以得出 N 与 n 的关系。对 t 提取公因式得到：

$$t = (\frac{1}{\theta} + n - 1)(1-\theta)^{n-1}(2\theta(1-\theta)^{n} - 1)\ln(1-\theta)$$

。其中 $(\frac{1}{\theta} + n - 1)(1-\theta)^{n-1}\ln(1-\theta) < 0$，而 $2\theta(1-\theta)^{n} - 1 < 2\theta(1-\theta) - 1 = -\theta^2 - (1-\theta)^2 < 0$，因此 $t > 0$，进而 $\frac{\partial N}{\partial n} > 0$。所以，当 θ 固定时，N 与 n 是正相关的。即在单个字节变动的情况下，如果模式匹配成功的概率固定，那么定位到的差异数据量与模式匹配窗口的长度成正比。

在定位差异数据时，上文已经证明差异数据中包含了所有变动数据，那么为了取得更好的定位效果，只需要降低差异数据的大小，即让 N 尽可能小。既然 N 与 n 是正相关的，且 n 为正整数，那么令 $n = 1$，代入公式(7)得到公式(9)。

由公式(9)可得：$N' = -\theta^{-2} - 1 < 0$。因此，在让 N 尽可能小的过程中，N 与 θ 是成反比的，即 θ 越大 N 越小。当 $\theta \to 1$ 时，字节变动对模式匹配结果没有影响，也就不会影响原有的分块结果，进而使得差异数据的定位量等于字节变动所在分块的大小。由上文可知，待分块数据的分块结果中，分块存在一个理论期望值，即 $E(x_i) = \frac{1}{\theta} + n - 1$，其中 x_i 为分块大小。当 $\theta \to 1$ 时，$E(x_i) = n$，即定位到的差异数据理论值为模式匹配窗口的大小。

理论上，根据公式(7)可以推导出 N 与 θ 的关系，但是经过较长时间的工作之后始终未能得出一个有效的结果。在后续的研究中将继续努力，以期可以得出 N 与 θ 的具体关系。

综上所述，在原数据只发生单个字节变动时，借助数据分块算法定位得到的差异数据的大小与数据分块算法的匹配窗口大小正相关。当匹配窗口大小为 1 时，差异数据的大小与窗口匹配成功的概率负相关。该结论可以通过图 5 形象地说明，图中 C_1 和 C_3 是分块中的非匹配窗口字段，C_2 和 C_4 是匹配窗口字段，深色背景表示因字节变动产生的差异数据。由上文的分析可知，在 θ 相同的情况下，C_1 长度与匹配窗口的长度无关，因此在图 5(b)场景下，差异定位的长度与匹配窗口的长度正相关。同理，在图 5(c)场景下，差异定位的长度与匹配窗口的长度也是正相关。

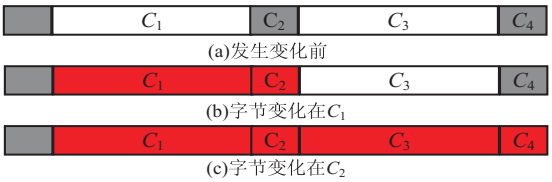

图 5 差异数据的大小与算法匹配窗口大小的关系

当原数据发生单个字节的增加或删除时，可以理解为双字节变成了单字节，或单字节变成了双字节，进而也可以得出类似的结论。对于多个字节发生变化的情况，可以理解成发生了多次单字节变化，进而也同样可以得出类似的结论。

因此可以得出结论：当原数据发生变化时，借助数据分块算法定位得到的差异数据的大小与数据分块算法的匹配窗口大小成正比。当匹配窗口大小为 1 时，差异数据的大小与窗口匹配成功的概率成反比。另外，当匹配窗口大小不为 1 时，需要从公式(7)中计算此时 N 与 θ 的关系。

需要注意的是，在实际使用中，当 n 固定时，θ 的值是无法取到任意值的，因为一个匹配窗口内的数据的可能性是有一定范围的。比如，一个长度为 1 的匹配窗口，那么窗口中数据的种类最多只有 256 种，无法依据这 256 种可能来设计任意的匹配概率。此外，当差异数据定位用在数据增量同步等场景时，对数据分块的数量有要求，分块数量的增多会产生更多的网络带宽。因此在定位差异数据时，并不是 θ 越大越好，应该先根据分块数量计算出大致的 θ 值，然后找出满足 θ 的最小 n 值，然后利用公式(7)来寻找最优的 θ，进而设计出一个较优的数据分块算法。

5 结束语

首先对借助数据分块算法来定位差异数据的过程进行抽象，利用数据集合的相关理论证明了这一过程的正确性，其中包括定位到的差异数据包含了所有的变化数据，原数据借助差异数据的帮助可以拼接得到变化后的数据。然后，将数据分块算法抽象为一种基于固定长度窗口的模式匹配过程，通过推导的方式得出了数据分块算法中窗口大小和匹配成功概率两个参

数与定位得到的差异数据大小之间的关系。并得出数据分块算法的窗口大小参数与定位得到的差异数据大小成正比,匹配成功概率参数在窗口大小为 1 的情况下与定位得到的差异数据大小成反比。这对设计数据分块算法有一定的参考意义。

不过,该文未能给出一个差异数据大小与匹配成功的概率之间具体的关系,此外 N 与 θ 的关系也未能给出一个准确的结论,有待进一步研究。

参考文献:

[1] TRIDGELL A. Effcient algorithms for sorting and synchronization[EB/OL]. 1999. https://www. samba. org/ ~ tridge/phd_thesis. pdf.

[2] LKHAGVASUREN I, SO J M, LEE J G, et al. Multi-level byte index chunking mechanism for file synchronization[J]. Int J Softw Eng Its Appl, 2014, 8: 39-50.

[3] MO R. Fingerprinting by random polynomials[EB/OL]. 1981. https://www. researchgate. net/publication/242608934_Fingerprinting_by_Random_Polynomials.

[4] XIA W, ZOU X, JIANG H, et al. The design of fast content-defined chunking for data deduplication based storage systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 31(9): 2017-2031.

[5] BJRNER N, BLASS A, GUREVICH Y. Content-dependent chunking for differential compression, the local maximum approach[J]. Journal of Computer and System Sciences, 2006, 76(3-4): 154-203.

[6] 张曙光,咸鹤群,王雅哲,等. 基于离线密钥分发的加密数据重复删除方法[J]. 软件学报,2018,29(7):1909-1921.

[7] 汪　帅,吕江花,汪滢鹤,等. 一种支持数据去冗和扩容的多媒体文件云存储系统实现[J]. 计算机研究与发展,2018,55(5):1034-1048.

[8] ZHOU B, JIN H, XIE X, et al. BBMC: a novel block level chunking algorithm for de-duplication backup system[J]. International Journal on Information, 2013, 16: 469-479.

[9] XIA W, ZHOU Y, JIANG H, et al. FastCDC: a fast and efficient content-defined chunking approach for data deduplication[C]//Proc. Usenix Atc 16 2016 Usenix Annu. Tech. Conf. Berkeley: USENIX ASSOC, 2016: 101-114.

[10] MOON Y C, JUNG H M, YOO C, et al. Data deduplication using dynamic chunking algorithm[J]. Lect Notes Comput Ence, 2012, 7654: 59-68.

[11] 孙继忠,马永强,李玉华. 基于字节指纹极值特征的数据分块算法[J]. 计算机工程,2010,36(8):69-70.

[12] ZHOU P, WANG Z, XIA W, et al. UltraCDC: a fast and stable content-defined chunking algorithm for deduplication-based backup storage systems[C]//2022 IEEE international performance, computing, and communications conference (IPCCC). Austin: IEEE, 2022: 298-304.

[13] XU Z, ZHANG W. QuickCDC: a quick content defined chunking algorithm based on jumping and dynamically adjusting mask bits[C]//2021 IEEE intl conf on parallel & distributed processing with applications, big data & cloud computing, sustainable computing & communications, social computing & networking (ISPA/BDCloud/SocialCom/SustainCom). New York: IEEE, 2021: 288-299.

[14] LU G, JIN Y, DU D H C. Frequency based chunking for data de-duplication[C]//2010 IEEE international symposium on modeling, analysis and simulation of computer and telecommunication systems. Miami Beach: IEEE, 2010: 287-296.

[15] KRISHNAPRASAD P K, NARAYAMPARAMBIL B A. A proposal for improving data deduplication with dual side fixed size chunking algorithm[C]//2013 third international conference on advances in computing and communications. Cochin: IEEE, 2013: 13-16.

[16] WU H, WANG C, LU K, et al. One size does not fit all: the case for chunking configuration in backup deduplication[C]//2018 18th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID). Washington D. C.: IEEE, 2018: 213-222.